REMARKS

Claims 1-3, 6-12, 14-19, and 21-23 are all the claims presently pending in the application. Claims 4, 5, 13, and 20 are canceled.

It is noted that the claim amendments are made only for more particularly pointing out the invention, and not for distinguishing the invention over the prior art. Further, Applicant specifically states that no amendment to any claim herein should be construed as a disclaimer of any interest in or right to an equivalent of any element or feature of the amended claim.

The Examiner's previous objection to the drawing for allegedly failing to show the feature described in the independent clams is understood as having been withdrawn.

The Examiner's previous objection that new matter was allegedly introduced in the previous amendment is also understood as having been withdrawn.

The Examiner's previous objection for claims 8, 12, 17, 18, and 20 for failing to define "BLAS" is also understood as having been withdrawn.

The Examiner newly objects to claim 23 for repeating claim 21. The amendment above revises claim 23 to depend from the apparatus claim 9, thereby rendering this point moot. Therefore, Applicants respectfully request that the Examiner reconsider and withdraw this rejection.

Claims 1-3, 6-12, 14-19, and 21-23 stand rejected under 35 U.S.C. § 112, second paragraph, as allegedly being indefinite. Claims 1-3, 6-12, 14-19, and 21-23 stand rejected under 35 U.S.C. § 101 as allegedly directed to non-statutory subject matter. Claims 1-3, 6-12, and 14-18 stand rejected under 35 U.S.C. § 102(b) as anticipated by U.S. Patent No. 6,357,041 to Pingali, et al. Claim 19 stands rejected under 35 U.S.C. § 103(a) as unpatentable over Pingali, further in view of non-patent literature by Philip Alpatov, et al., "PLAPACK: Parallel Linear Algebra Package Design Overview" (hereinafter, "PLAPACK"), and claims 21-23 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Pingali, further in view of US Patent 5,099,447 to Myszewski.

These rejections are respectfully traversed in the following discussion.

I. THE CLAIMED INVENTION

The claimed invention, as exemplarily defined in independent claim 1, is directed to a method of improving at least one of speed and efficiency when executing a linear algebra subroutine on a computer having a memory hierarchical structure including at least one cache. For a level 3 matrix multiplication processing, it is determined which matrix will have data for a submatrix block residing in a lower level cache of the computer and which two matrices will have data for submatrix blocks residing in at least one higher level cache or a memory. The data from the selected two matrices is streamed for the executing of the level 3 matrix multiplication processing.

The present inventors have recognized that conventional linear algebra processing based on LAPACK subroutines, for example, are not optimal.

The claimed invention, on the other hand, along with various other techniques described in the co-pending applications, provides techniques that improve processing efficiency. More specifically, the present invention provides a memory management method allowing for a streaming of data through a cache, using another operand as having the "matrix role" and being resident in the cache.

II. THE REJECTION UNDER 35 USC §112, SECOND PARAGRAPH

The Examiner rejects to claims 1-3, 6-12, 14-19, and 21-23 for allegedly being indefinite. The Examiner considers it to be unclear how the determination step of the independent step relates to other limitations of the claimed invention, since "*... the multiplication processing only needs the data from higher level cache as L2.*"

In response, Applicants point out that there are three matrices involved in a level 3 matrix multiplication. Claim 1 describes that one of the matrices has its working data block resident in the lower level cache and data from working blocks of the other two matrices reside in the next higher level cache or memory.

Applicants respectfully direct the Examiner's attention to the discussion at line 19 of page 9 through line 2 of page 10, wherein is mentioned that L3 would contain at least part of the three data components of the matrix multiplication: size M3xK3 piece of matrix A; size K3 x N2 piece of matrix B; and size M2 x N2 piece of matrix C (refer to Figure 5).

As explained at line 19 of page 15 through line 16 of page 17, the present invention teaches (see lines 11-12 on page 16) that, contrary to conventional wisdom that considers that there is only one kernel for matrix multiplication (e.g., executing the 3-variable DO loop occurring at the lowest level in the FPU), there are six possible alternative DGEMM kernels that could perform this subroutine and one will be most efficient to perform this DO loop processing, as based on which of the three matrix blocks will fit into L1cache and which will then stream down through L2 cache.

Therefore, Applicants respectfully request that the Examiner reconsider and withdraw this rejection.

III. THE 35 USC §101 REJECTION

Claim 1 is useful and practical. It does so, by changing how the matrix is stored in the memory of the computer. The Examiner states in paragraph 7: "*Claim 1 merely discloses steps/components for performing matrix multiplication ....*"

This statement is incorrect. How the matrix data gets into and out of the caches greatly effects how efficient the matrix subroutine will execute. It is implicit in claim 1 that the matrix is stored in one of the two standard formats of DLA.

In the paper "High-performance linear algebra algorithms using new generalized data structures for matrices", by co-inventor Fred Gustavson, it is shown that these standard data structures hurt the performance of matrix subroutines. These results are generally accepted by the Engineering and Scientific Community at large.

Broadly speaking, Claim 1 describes a better data structure for matrices that the DGEMM subroutine will be processing. The result will be a more efficient execution of the DGEMM via the streaming process. Streaming will only work well for these better data structures.

The Examiner continues to reject claims 1-3, 6-12, 14-19, and 21-23 under 35 U.S.C. §101. The Examiner characterizes that these claims:

"*... merely disclose steps of streaming data from cache without further disclosing a practical/physical application or a useful and tangible result since the claims appear to preempt every substantial practical application of the idea embodied by the claim and there is no cited limitation in the claims that breathes sufficient life and meaning into the*

preamble so as to limit it to a particular practical application rather than being so broad and sweeping as to cover every substantial practical application of the idea embodied therein."

In response, Applicants respectfully point out that the Examiner's above-recited rationale contains various points of confusion, as follows:

First, the Examiner mischaracterizes the claimed invention. The independent claims actually define the preliminary steps for setting up of the machine to accommodate more efficient and/or faster execution of matrix multiplication processing, by determining which matrix will have its data block resident in the L1 cache and which matrices will have data streaming through L1 cache from higher levels. The conventional method for matrix multiplication simply invokes the single matrix multiplication kernel that is available, using data conventionally stored in memory and makes no attempt to determine how best to place the matrix data in the cache hierarchy for purpose of executing the matrix multiplication processing. Thus, the result/use of the present invention is that of initially setting up the machine for optimal matrix multiplication processing that will then be subsequently executed on the machine.

Contrary to the Examiner's characterization, the streaming described in the second claim limitation of the independent claims occurs only after the machine has been set up by completing the first step of having determined how the matrix data will reside in the cache hierarchical structure. As clearly stated in the first claim limitation, one matrix will be selected to have its working sub-block residing in L1 cache and data from the remaining two matrices will stream in through L1 cache from higher cache levels.

Second, it is brought to the Examiner's attention that execution of matrix multiplication processing without first executed this preliminary setup procedure as defined by the independent claims will not infringe the claimed invention, even if streaming is used.

Therefore, contrary to the Examiner's characterization, the present invention is not preempting the conventional method of matrix multiplication that does not recognize the problem being addressed by the present invention (e.g., that there is a preliminary setup procedure that will optimize the subsequent matrix multiplication processing) and is not even preempting the streaming of matrix data that is done in any other manner or without this initial determination procedure.

Therefore, it is this preliminary setup procedure that is significantly of interest in the present invention and the issue of preemption, not the subsequent matrix multiplication processing itself. Moreover, matrix data streaming, by itself, and/or in different format from that described in the independent claims are not being claimed or preempted by the present invention. The real-world application for which the matrix multiplication is being executed is completely irrelevant to this preliminary startup procedure and to the present invention. That is, it does not matter whether the matrix multiplication is being used for a solving a scientific or engineering problem or is being used in a game or graphics rendering application (see lines 13-16 of page 3 of the specification).

Thus, contrary to the Examiner's characterization above, as long as the preliminary optimization step of determining the location of matrix data in the cache hierarchical structure is not executed as a setup procedure, the present invention is not preempting matrix multiplication or the streaming of matrix data and is not, therefore, preempting any and all possible applications of matrix multiplication.

Moreover, the Examiner's position that the claimed invention articulate the practical uses for which the invention would be confined is not consistent with the holding in *State Street*:

"*Today, we hold that the transformation of data, representing discrete dollar amounts, by a machine through a series of mathematical calculations into a final share price, constitutes a practical application of a mathematical algorithm, formula, or calculation, because it produces "a useful, concrete and tangible result" – a final share price momentarily fixed for recording and reporting purposes and even accepted and relied upon by regulatory authorities and in subsequent trades.*"

Thus, the State Street Court held that this "transformation of data" to provide a final share price by itself produced a useful, concrete and tangible result (e.g., a final fixed share price) and that it was not necessary to define in the claim that this result would be used for "recording purposes" or "reporting purposes" or "accepted and relied upon by regulatory authorities" or "accepted and relied upon in subsequent trades."

Therefore, Applicants submit that the above-recited wording from the *State Street* holding clearly indicates that it is not necessary to describe in the claims of the present invention the ultimate use for which the matrix multiplication processing is applied.

Moreover, taking the State Street description as an example, Applicants submit that

the result in the present invention is the determination of which matrix data will reside in L1 cache and which matrix data will reside at a higher level for streaming and the use of the claimed invention is the improvement of efficiency and/or speed of the matrix multiplication processing.

Applicants further submit that the claimed invention not only defines the useful, concrete and tangible result (e.g., determine relative location of the matrix data) but also defines the use (improvement of efficiency/speed) of that result (since the determination of the relative locations is used to improve efficiency/speed of the subsequent matrix multiplication processing).

Applicants believe that the statutory subject matter rejection of record improperly attempts to arbitrarily force Applicants to identify in the claims "uses" that are not only irrelevant to the significance of the invention but also not required as a matter of law, based on *State Street*.

In view of the foregoing, the Examiner is respectfully requested to reconsider and withdraw this rejection.

IV. THE PRIOR ART REJECTIONS

The Examiner alleges that Pingali anticipates claims 1-3, 6-12, and 14-18, and, when modified by PLAPACK, renders obvious claim 19, and, when modified by Myszewski, renders obvious claim 21-23.

Applicants respectfully disagree.

Although newly-cited Pingali addresses the same problem of improving efficiency in matrix manipulation, including matrix multiplication, its approach does not satisfy the plain meaning of the claim language of even the independent claims, for several reasons.

First, Pingali addresses a machine having a single layer of cache and teaches the different technique of bringing blocks of data from all the matrices into this cache (e.g., line 65 of column 2 through line 2 of column 3). It further teaches to determine how to utilize this data as much as possible by executing any processing that touches these stored blocks in cache. All compilers represent sub matrices as well as matrices in standard format. Pingalli's invention is therefore sub-optimal.

Thus, in contrast to the claimed invention, there is no step in Pingali to first

determine which matrix should have its block reside in L1 and which blocks will reside in higher levels of cache or memory and be streamed downward through L1 cache into the FPU (or CPU, as done in Pingali. Even if Pingali did so his block would be in standard format and his results would be sub-optimal.

In the rejection for claims 1-3, 6-12, and 14-18, relative to the first claim limitation of the independent claims, the Examiner points to column 3, lines 8-28, and column 6, lines 40-68. However, these lines merely describe that the matrices are divided into blocks and the order that the blocks of data are to be brought into cache is determined. It is also noted that the blocks of Pingali and the blocks of the present invention are different.

This description fails to satisfy the plain meaning of the language of the first claim limitation requiring a determination of which of the matrices will have its working block of data to be resident in the lowest level of cache and which matrices will have their working blocks resident at higher levels of cache or in memory and, again, Pingali's blocks are different from the blocks of the present invention.

The Examiner does not rely on secondary references PLAPACK and Myszewski to overcome this fundamental deficiency of primary reference Pingali, so neither of these two secondary references overcomes this deficiency of Pingali.

As clearly explained in lines 4-7 of page 18 of the specification:

"*Thus, to be more specific, in the context of the present invention, one is using streaming where the vector and scalar parts of matrices fit in the L2 (not L1) cache. In other words, only the submatrix playing the role of the matrix is L1 cache-resident.*"

Hence, turning to the clear language of the claims, in Pingali there is no teaching or suggestion of: "... determining, for a level 3 matrix multiplication processing, which matrix will have data for a submatrix block residing in a lower level cache of said computer and which two matrices will have data for submatrix blocks residing in at least one higher level cache or a memory ....", as required by independent claim 1. The remaining independent claims have similar wording.

For this reason alone, Applicants respectfully submit that all pending claims are clearly patentable over Pingali, and the Examiner is respectfully requested to reconsider and withdraw these rejections based on Pingali.

Moreover, it is brought to the Examiner's attention that the matrices that Pingali deals with are in standard format. Streaming does not work or works poorly for standard

format. In 1999, when Pingali submitted his patent, many, if not all, computers lacked a streaming capability. Therefore, Pingali fails to address the present invention involving streaming capability.

Relative to the rejection based upon secondary reference PLAPACK, this reference also uses standard formats, as Pingali must also use, for which streaming will not work or works poorly. Pingali provides compiler techniques which libraries such as PLAPACK might use to compile their source codes, but the independent claims of the present invention show a way to improve the ideas of Pingali and the performance of PLAPACK. Therefore, secondary reference PLAPACK does not render obvious either the independent claims or claim 19.

Relative to the rejection for claims 21-23, based upon secondary reference Myszewski, Applicants respectfully point out that Myszewski, consistent with conventional wisdom, only considers standard format, as well as only one kernel. Figure 5 of Myszewski details a specific example of a 3 by 3 matrix and how the individual elements of the matrix are used in his invention. Figure 5 shows the selecting of matrix elements.

In contrast, the present invention is selecting processing routines (kernels). Since there does not seem to be a corresponding concept in Myszewski, the Examiner is respectfully requested to point to a specific location in Myszewski that describe selection one of six alternative kernels or subroutines.

In the rejection of record, the Examiner points to lines 55-64 of column 4, lines 34-55 of column 14, lines 15-18 of column 15.

Lines 55-64 of column 4 are as follows:

"*It should be noted that the decision to divide the shaded areas 10, 16 into four blocks is for the purposes of illustration, and that the matrix could be divided into a number of block sizes. In practice, the optimum size of the matrix blocks will depend on computer system parameters such as cache size, and tradeoffs to be discussed later on. It should also be noted that the width of the blocks in the first term matrix should be the same as the height of the corresponding blocks in the second term matrix.*"

Lines 34-55 of column 14 read as follows:

The assembly language program presented in appen-
dix A generally follows the above-described flowchart, and it may be consulted for additional implementational details.
It is noted that, while the subroutine domine performs a deterministic,

static allocation of blocks to processors,
40 other allocations are possible, for example, dynamic self-scheduling, in which a processor completing a block selects the next block to be completed.
It is also noted that the flowchart and code for mu8blk allow for either or both of the term matrices A
45 and B to be transposed. This is accomplished by performing a different' initialization for the info parameter block. The increments found at elements 3, 4, 7, 8, 14 and ***15*** *of the info block are adjusted accordingly.*
It is further noted that this method can be used in
50 conjunction with Strassen's technique or another asymptotic complexity reduction method, for multiplying large matrices. In this case, the technique would be applied
to the matrix recursively until it was no longer efficient to do so. The resulting sub-matrices could then
55 be tackled using the above-described procedure.

Columns 15-16 read as follows:

```
                                                                    15                                                                  16
      recursive subroutine domine ( iproc, nproc, lda, a, b, c,                                                                        n, m )

      Each processor          (given by iproc ) in an Alliant complex of size nproc
      runs this subroutine to determine and do its share of blocks of a matrix multiply-add c - c + a*b, where
      all share the leading dimension lda, a is n by m, b is m by n, and c is therefore n by n.

      •   RESTRICTIONS:   n is an integer multiple of m
      c                   m is 20

      c Input arguments:
      c     iproc     *. current processor # (numbered from 0)
      c     nproc       total processors
      c     lda         leading dimension of a, b, and c
      c     n           number of rows of a, columns of b, and rows of c
      c     m           number of columns of a, rows of b, and columns of c
      c     a         - matrix of leading dimension lda and dimension n by m
      c     b         •• matrix of leading dimension lda and dimension m by n

      c • Input/Output arguments:
      c     c                                   matrix of leading dimension lda and dimension n by n
      c
      c Local cacheable variables used by mu8blk
      c           bcache, ccache, info

      c • External routines:
                                          c     mu81d:      (to be inlined) simple fortran routine to isolate details of info
                                                c     mu8blk:     (the magic) cache pipelined 20 x 20
                                                block processor
c •  Initialize i and j
     i ■ m*mod(myvov,nb)                        real*8          a(0:1da-
     j ■ m*(myvov/nb)                           1,0:*),b(0:1da-1,0:*),c(0:1da-
       print 98, iproc,myvnv, myvov, i,j        1,0:*)
                                                real*8 bcache(0:m-
                1,2),ccache(0:m-1,0:m-1,2) integer info(24)

      c nb = number of blocks on a side
      c nb2         total number
      c num         minimum number each processor must take
      c left - number of processors that must take an extra block nb  n/m
              nb2       nb*nb
              num - nb2/nproc
              left          nb2-num*nproc

              call initinfo(info,lda,m,bcache,ccache)

      c • Calculate the number of blocks to do (myvnv) and the first one to do (myvov) if      (iproc.lt.left) then
        myvnv num+1
      else
```

```
  myvnv - num
endif
if              (iproc.le.left) then
  myvov                 iproc*(num+l)
else
  myvov                                 left*(num+1) + (iproc-left)*num
endif
if                      (myvnv.eq.0) return




c ● Step 1: load first a, b, and c and push ----------

        call mu8ld ( info, loc(a(i3O)), loc(b(0,j)), loc(c(i,j)) ) print 99, iproc,i,j,(k,info(k),
        k■1,18)
             format
       &  (' Processor I ',i2,' Running block i,j ■ ',2i4,' Info:'
          /9(2(i3':',i16)/) )
        call mu8blk ( info )                                                    ----------
c ● If there is only one block to do: push the pipe to advance it if (myvnv .eq. 1) then
            call muald ( info, 0, 0, 0 )
        print 99, iproc,i,j,(k,info(k), k■1,18)
            call mu8blk ( info )
```

Applicants note that one having ordinary skill in the art would consider that lines 55-64 of column 4 above talk about selecting cache block sizes, lines 34-55 of column 14 above describe programming details, and columns 15-18 above describe the beginning of subroutine dmm.

Therefore, Applicants respectfully submit that the above recitations from Myszewski do not reasonably describe selecting one of six possible kernels. Applicants respectfully submit that the Examiner's characterization in paragraph 12 of the Office Action does not seem to be supported in Myszewski, and the Examiner is requested to be more specific in his allegation of this description of the "processor switches back and forth between said two selected kernels as streaming data transverses said different levels of cache".

Finally, it is again noted that Myszewski was conceived before January 22, 1990, when M = 1. That is, during that period, there was memory and M = 1 cache.

As explained at lines 12-18 of page 14 of the specification:

"*The importance of having six kernel types available is that stride one memory access is desirable for matrix processing. The matrices A and B are usually stored either by row or by column. By having six kernel types, one can choose a kernel in which stride*

one is available for both operands. Having only one kernel type, instead of six kernel types, means that data copy must be done to provide the format of the one conventional kernel. This means a certain performance loss that might have to be repeated several times."

Therefore, Applicants submit that there are elements of the claimed invention that are not taught or suggest by Pingali, PLAPACK, or Myszewski. Therefore, the Examiner is respectfully requested to withdraw these rejections.

V. FORMAL MATTERS AND CONCLUSION

In view of the foregoing, Applicants submit that claims 1-3, 6-12, 14-19, and 21-23, all the claims presently pending in the application, are patentably distinct over the prior art of record and are in condition for allowance. The Examiner is respectfully requested to pass the above application to issue at the earliest possible time.

Should the Examiner find the application to be other than in condition for allowance, the Examiner is requested to contact the undersigned at the local telephone number listed below to discuss any other changes deemed necessary in a telephonic or personal interview.

The Commissioner is hereby authorized to charge any deficiency in fees or to credit any overpayment in fees to Assignee's Deposit Account No. 50-0510.

Respectfully Submitted,

Date: January 4, 2008

Frederick E. Cooperrider
Registration No. 36,769

McGinn Intellectual Property Law Group, PLLC
8321 Old Courthouse Road, Suite 200
Vienna, VA 22182-3817
(703) 761-4100
Customer No. 21254